

Метрики качества для регулярных нагрузочных тестов

Автоматизация остановки
неуспешных тестов •



О себе

Цитман Дмитрий



Более 10 лет

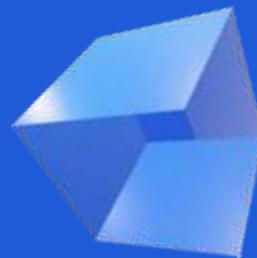
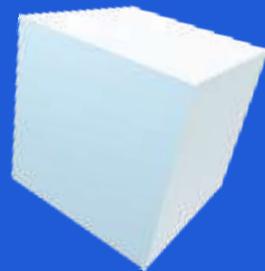
в области нефункционального
тестирования



Занимаюсь разработкой
и нефункциональными видами
тестирования



Предпосылки оценки качества

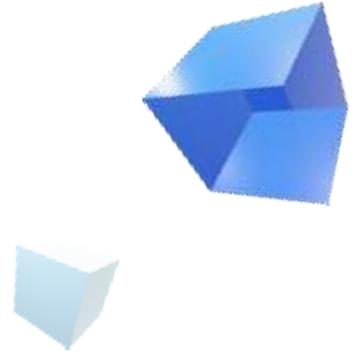


Базис качества

1. Воспроизводимость
2. Скорость
3. Доступность
4. Однозначность

Обоснования

1. Требования к компетенциям
2. Стоимость исправления дефектов
3. Ценность найденных дефектов
4. Прозрачность в принятии решений



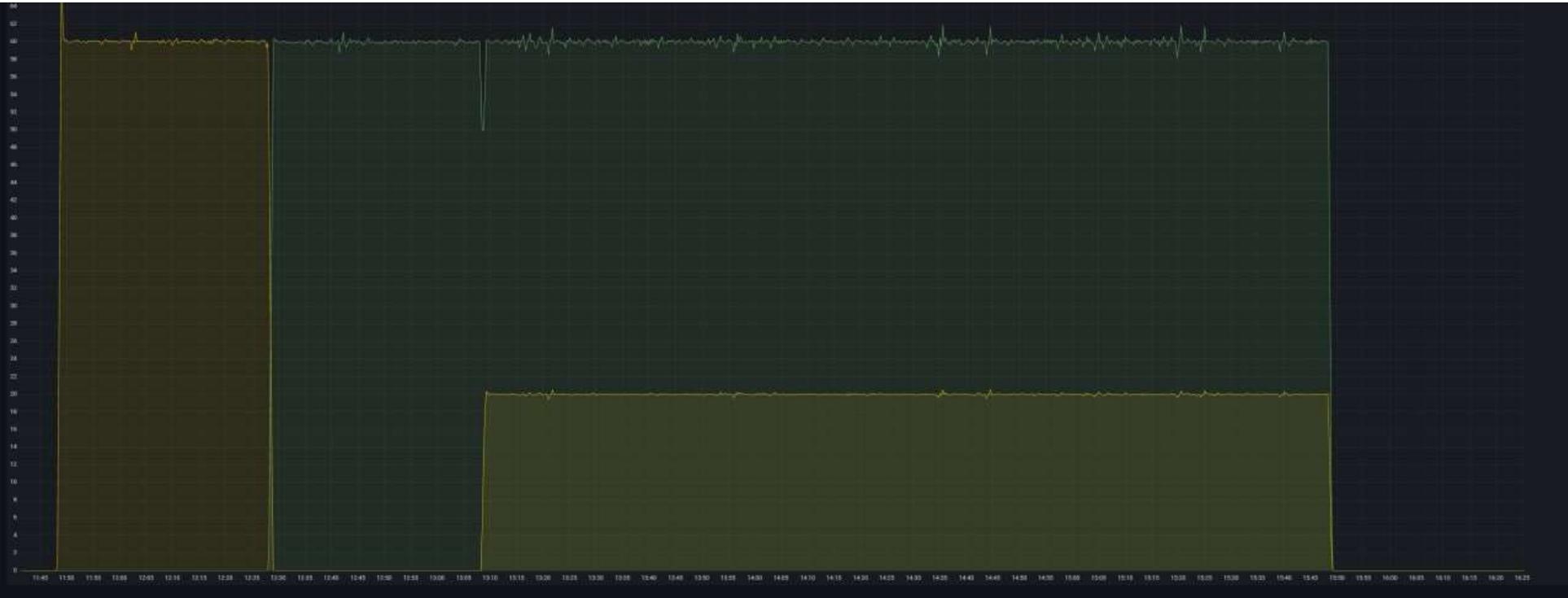
Неуспешные тесты

1. Тратят много времени
2. Требуют сложных решений
3. Не всегда воспроизводятся

Пример теста



Успешный тест

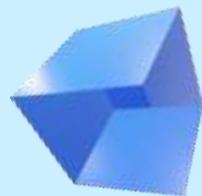


Неуспешный тест



ДОВОДЫ К ОСТАНОВКЕ

1. Конвейер релизов остановлен
2. Данных достаточно
3. Переиспользование ресурсов



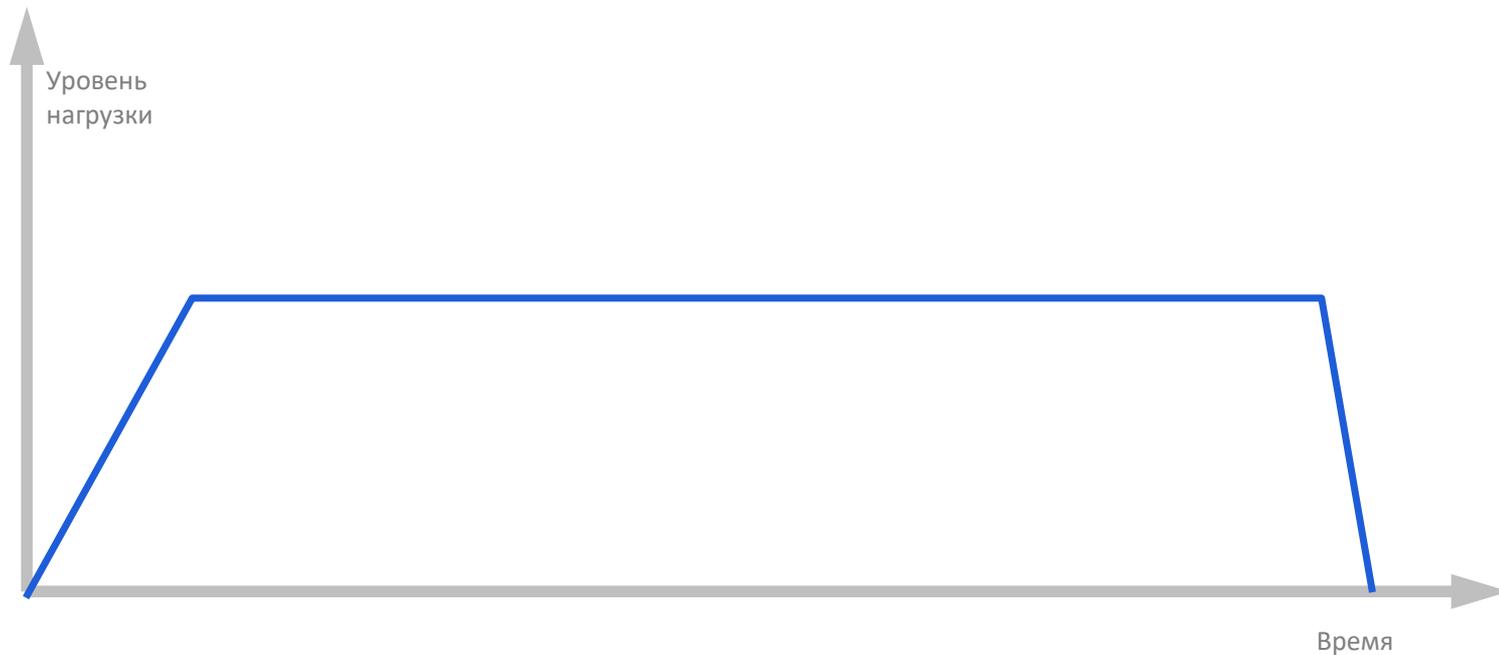


СЛОЖНОСТИ

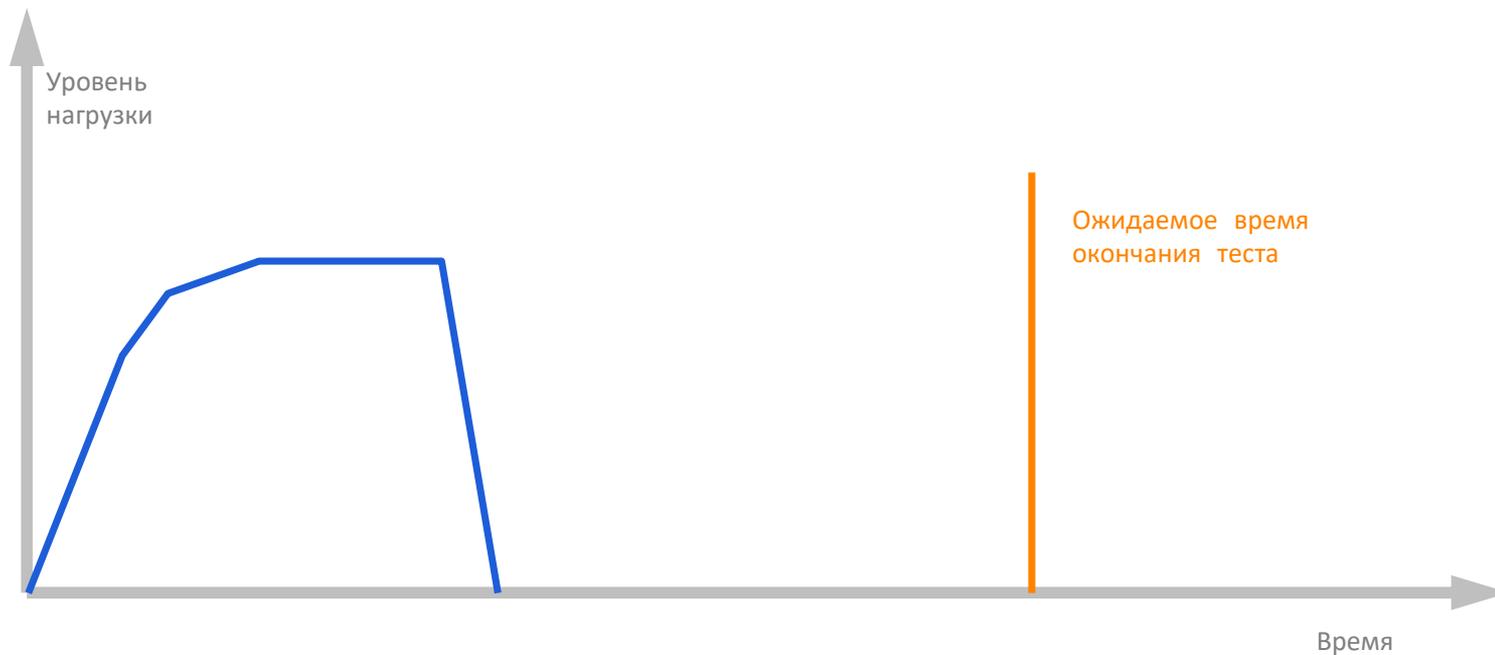
с оценкой состояния

1. Вариации в поведении функционала
2. Случайные события
3. Наличие артефактов поведения
4. Стабильность испытательного контура

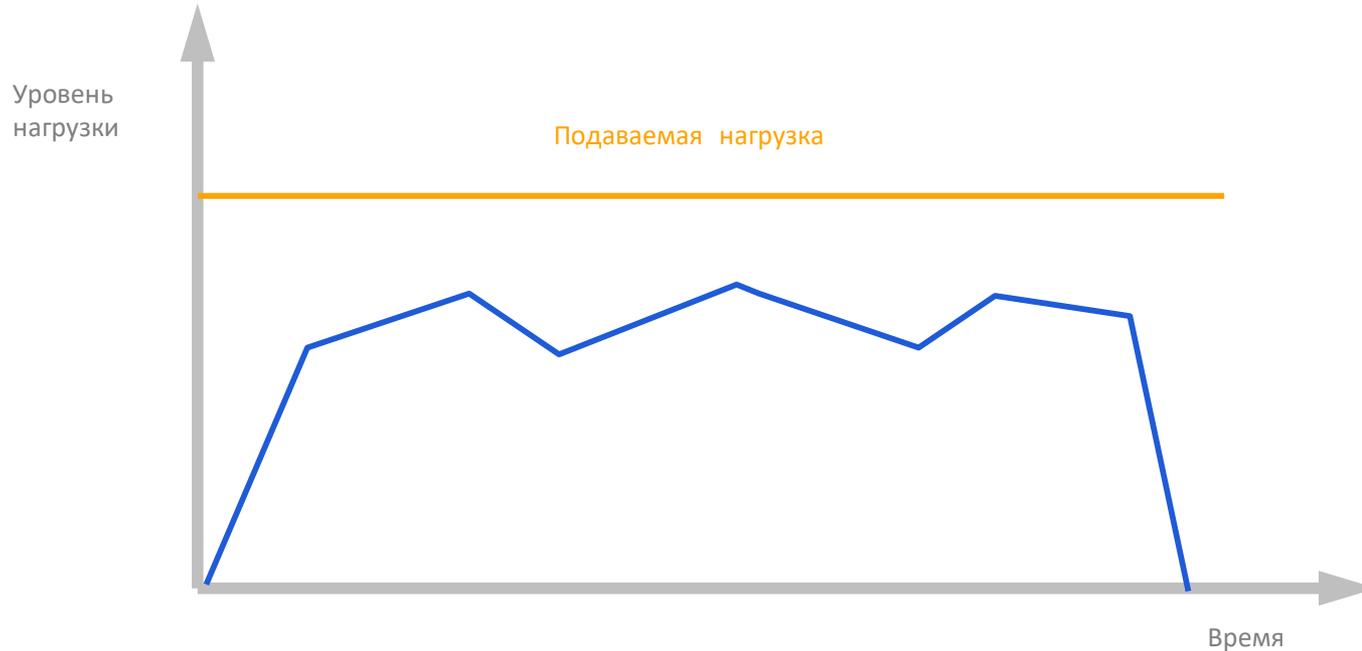
Идеальное поведение



Идеальный неуспешный тест



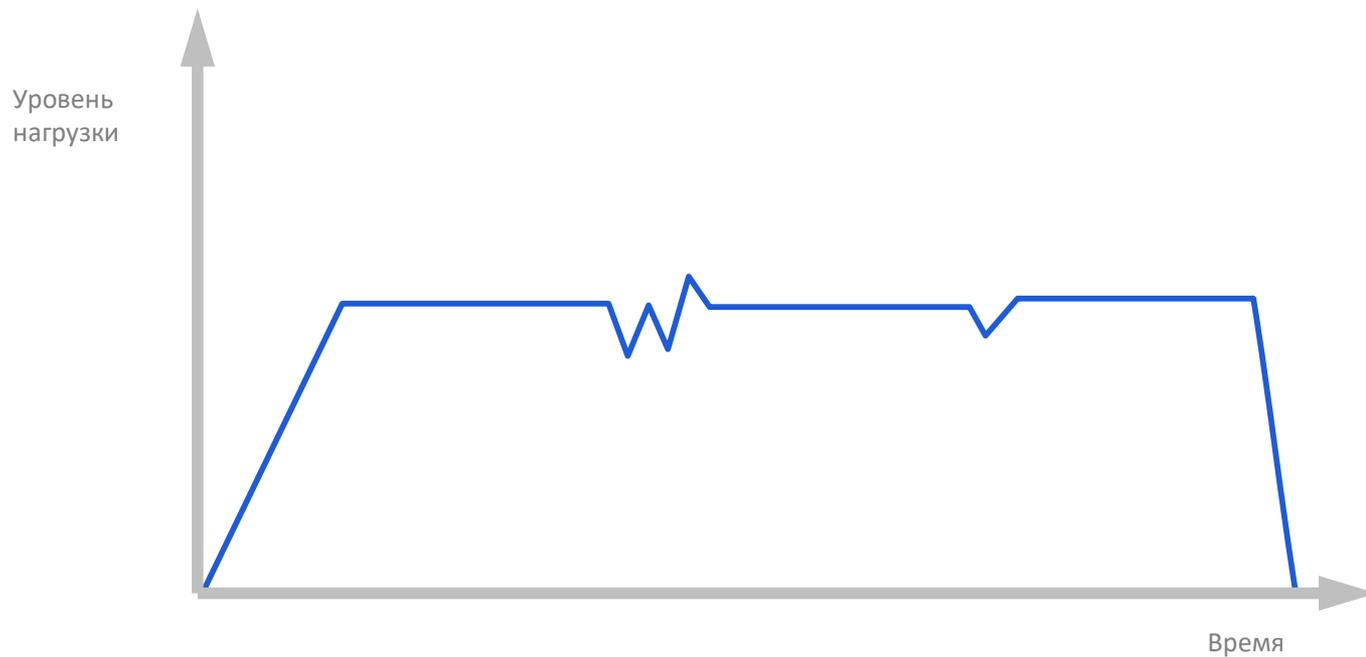
Несоответствие требованиям на протяжении всего теста



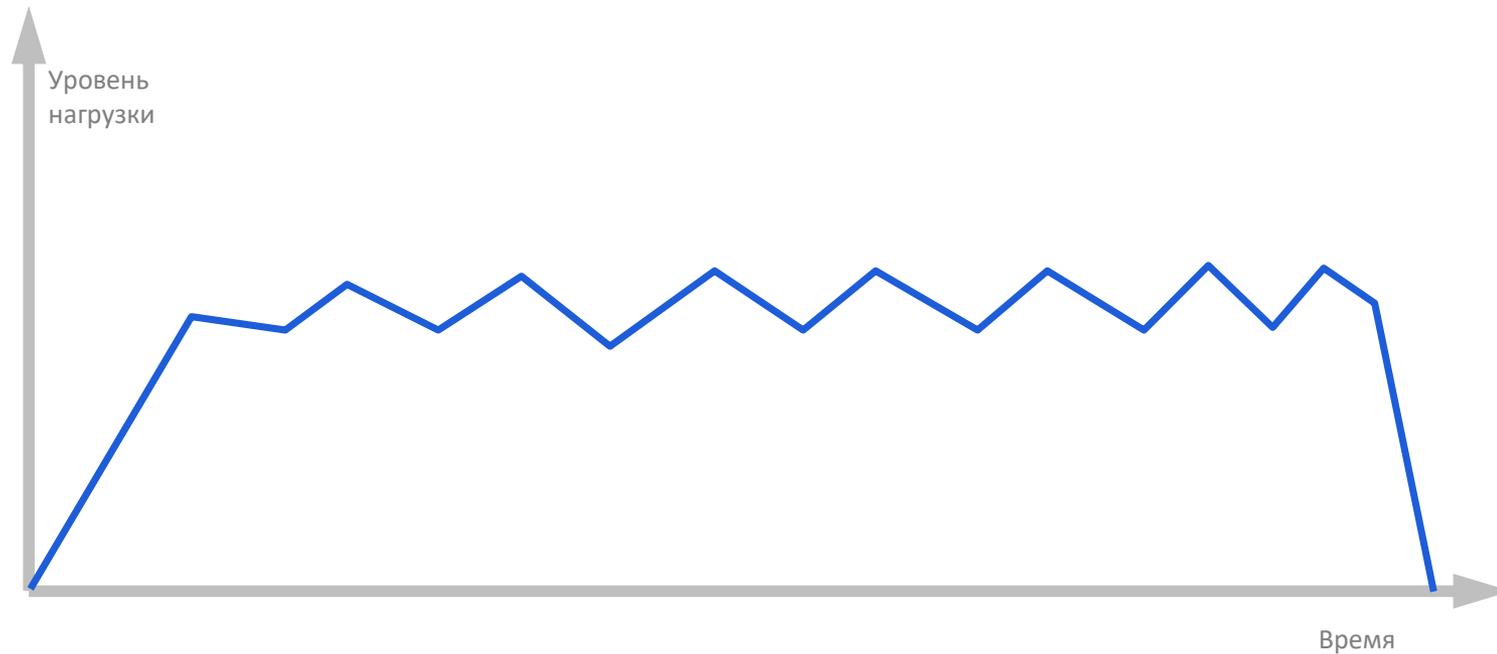
Единичные несоответствия требованиям



Множественные несоответствия

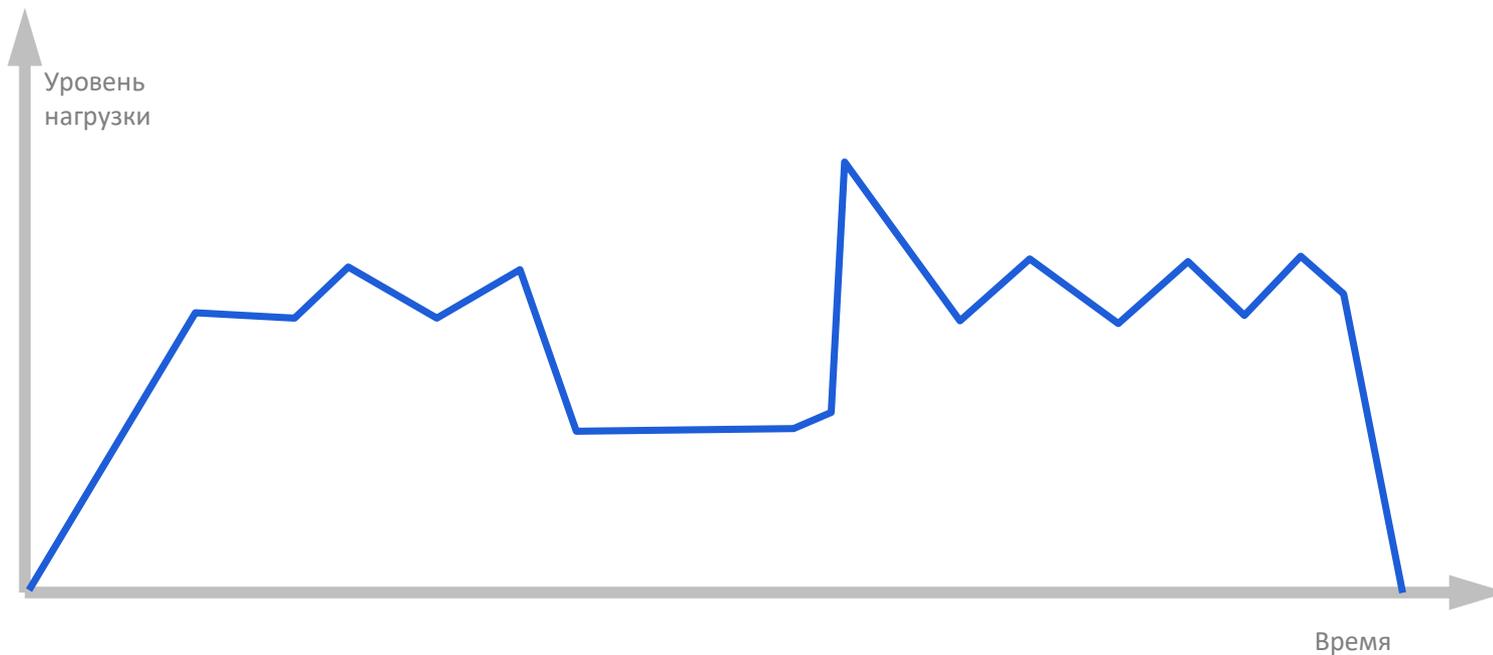


Колебания



Глубокое проседание

и восстановление



Оценка

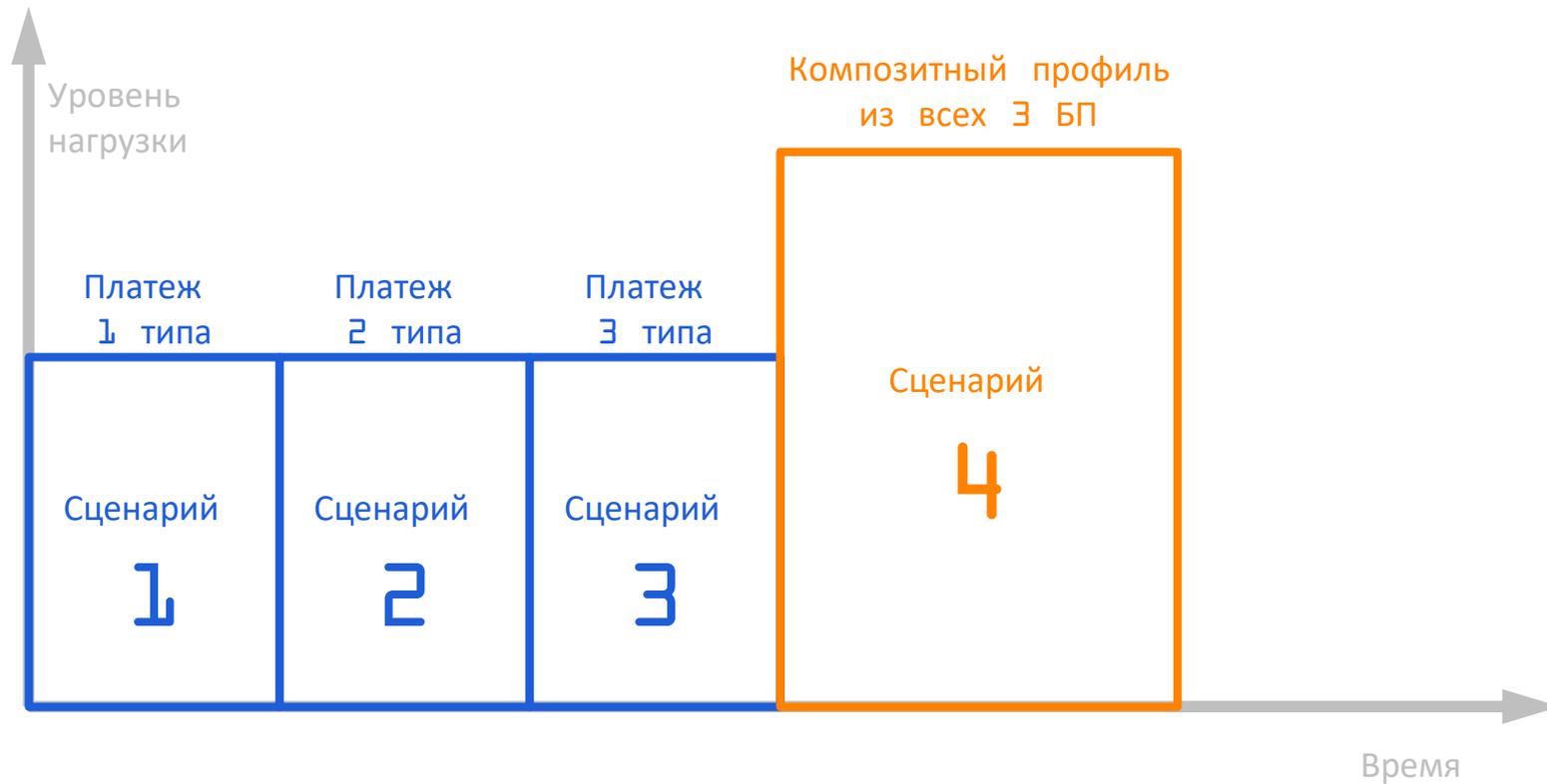
отклонений поведения

1. Вероятность
2. Критичность
3. Возможность исправления
4. Масштаб



Модель тестирования

1. Тривиализация профиля НТ
2. Фиксация уровня нагрузки



Преимущества

КОМПОЗИТНЫХ сценариев

1. Последовательная проверка БП
2. Редкое изменение профиля
3. Персональные требования
4. Однозначность выводов

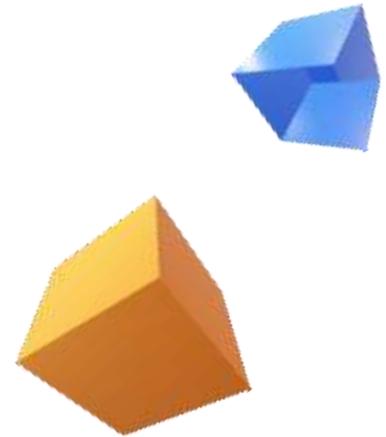
Параметры

длительности шагов

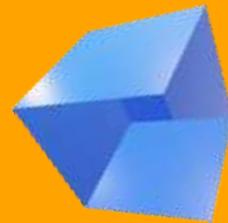
1. Времени обработки
2. Уровню нагрузки
3. Стабильность поведения
4. Сложность сценария

Длительность шагов

1. 20 минут на простые сценарии
2. 1 час на сложные сценарии



Воспроизводимость и однозначность



Невозможно

оценивать результаты без
истории запусков

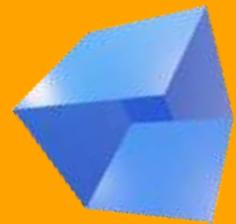
Регулярное тестирование

1. Набранная история
2. Итеративные изменения

Регрессионный нагрузочный тест

1. Регулярен
2. Обязателен
3. Проводится до 5 раз в день

Можно ли выносить
релиз?

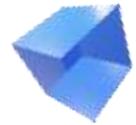
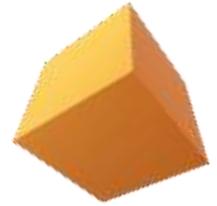


Регресс

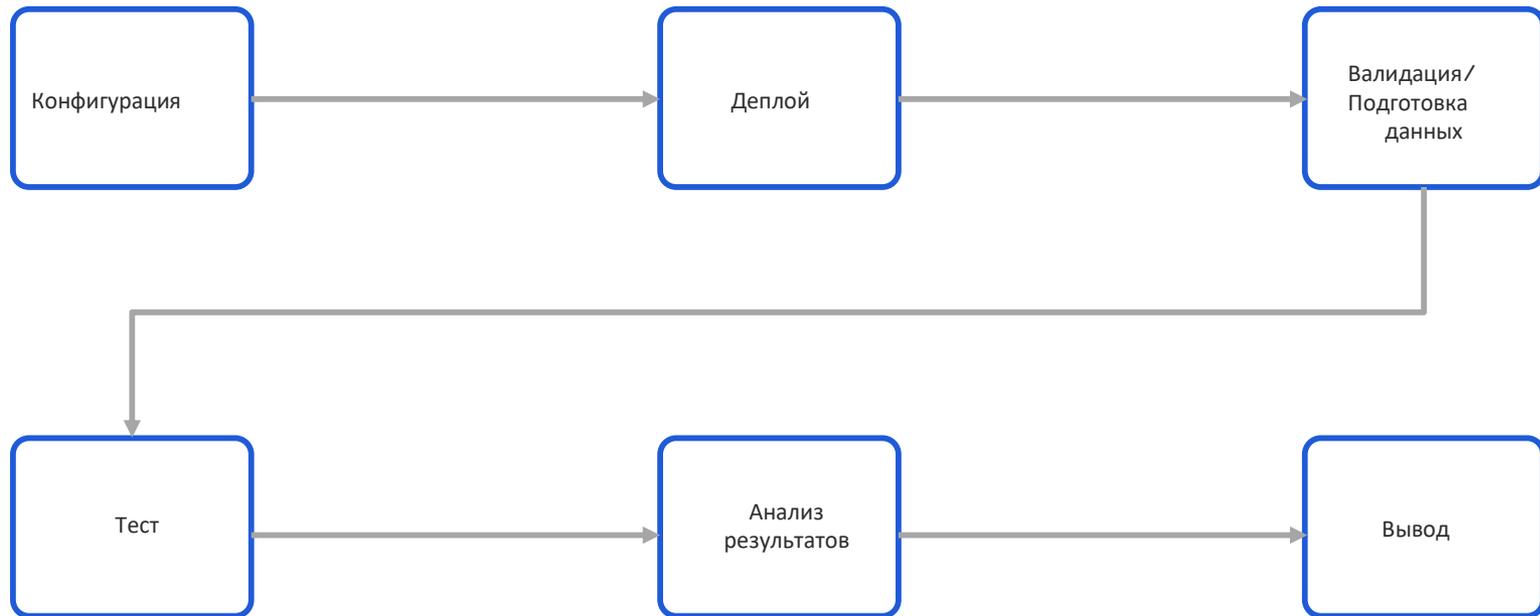
1. Не занимается поиском максимума
2. Не проверяет стабильность
3. Не ищет утечки
4. Не занимается поиском узких мест
5. Не сравнивает релизы

Другие тесты - если есть

1. Изменение архитектуры
2. Модели поведения
3. Добавление нового функционала



Этапы проведения регресса



85% времени

Підходить

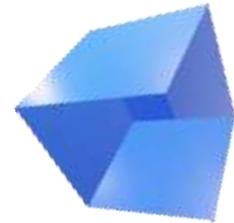
для автоматизації
остановки

Критерии успешности регресса НТ

1. Соответствие модели НТ
2. Соответствие системы требованиям
3. Фиксация результатов

Статистики

1. Перцентили
2. Среднее
3. Границы

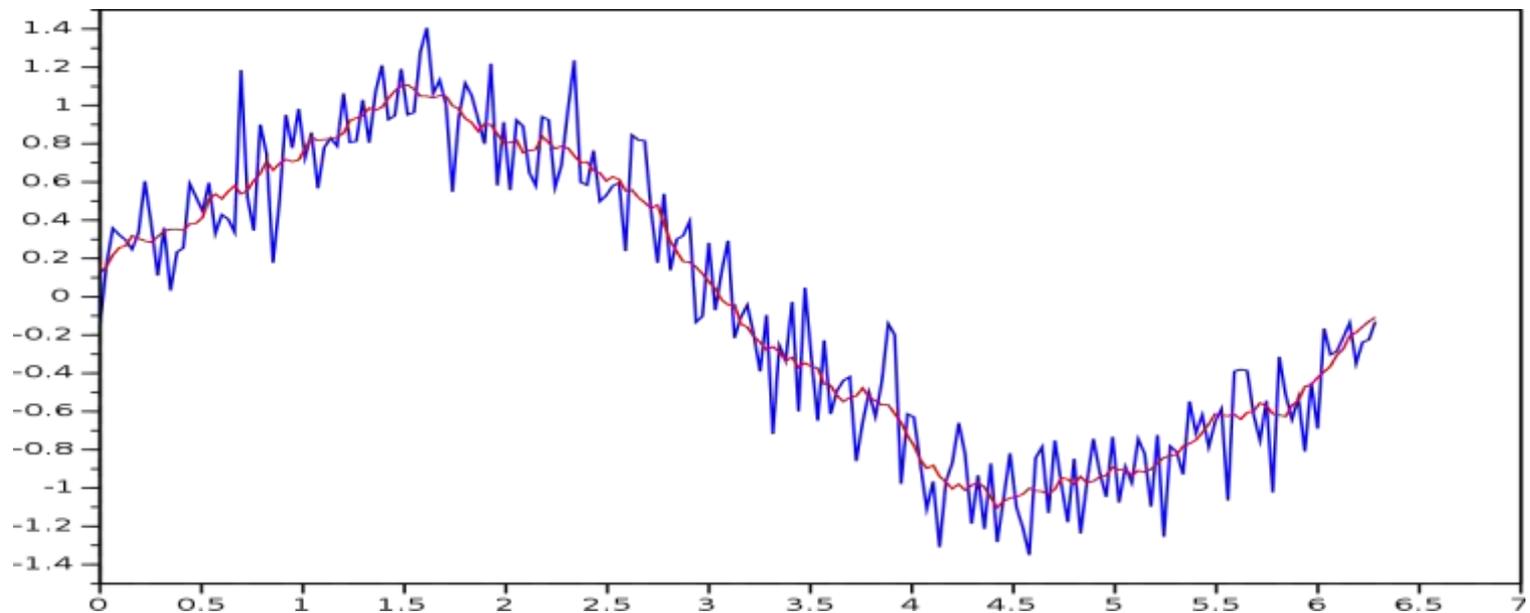


Сбор статистик

1. За период
2. За весь тест

Скользящее

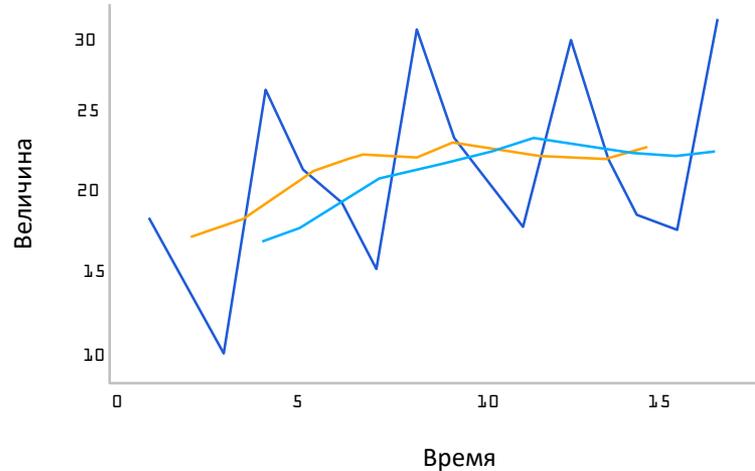
значение



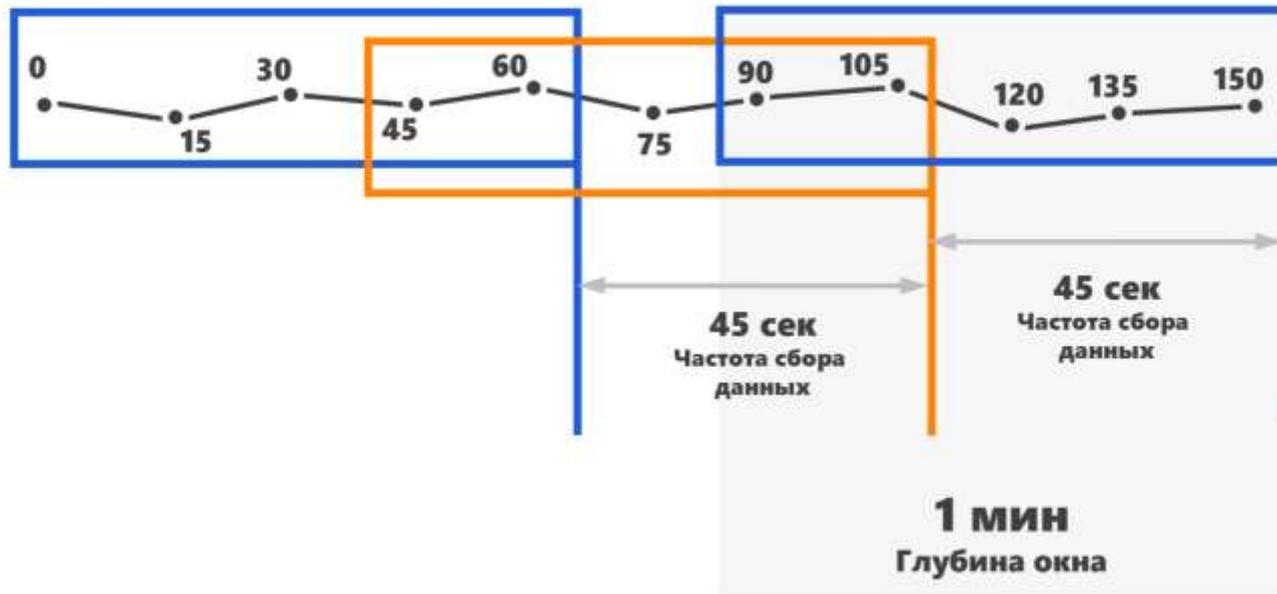
Скольльзящее

значение

$$\hat{y}_t = \frac{1}{p} \sum_{j=t-m}^{t+m} y_j$$

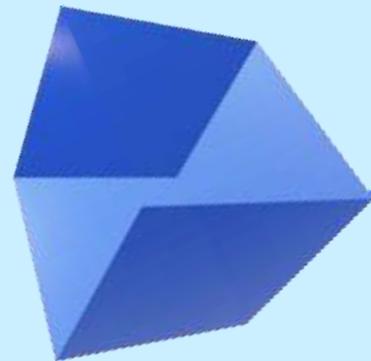


Принцип работы





Требования



Версионирование требований

1. История требований
2. История эталонных тестов
3. Привязка результатов к конкретной версии требований



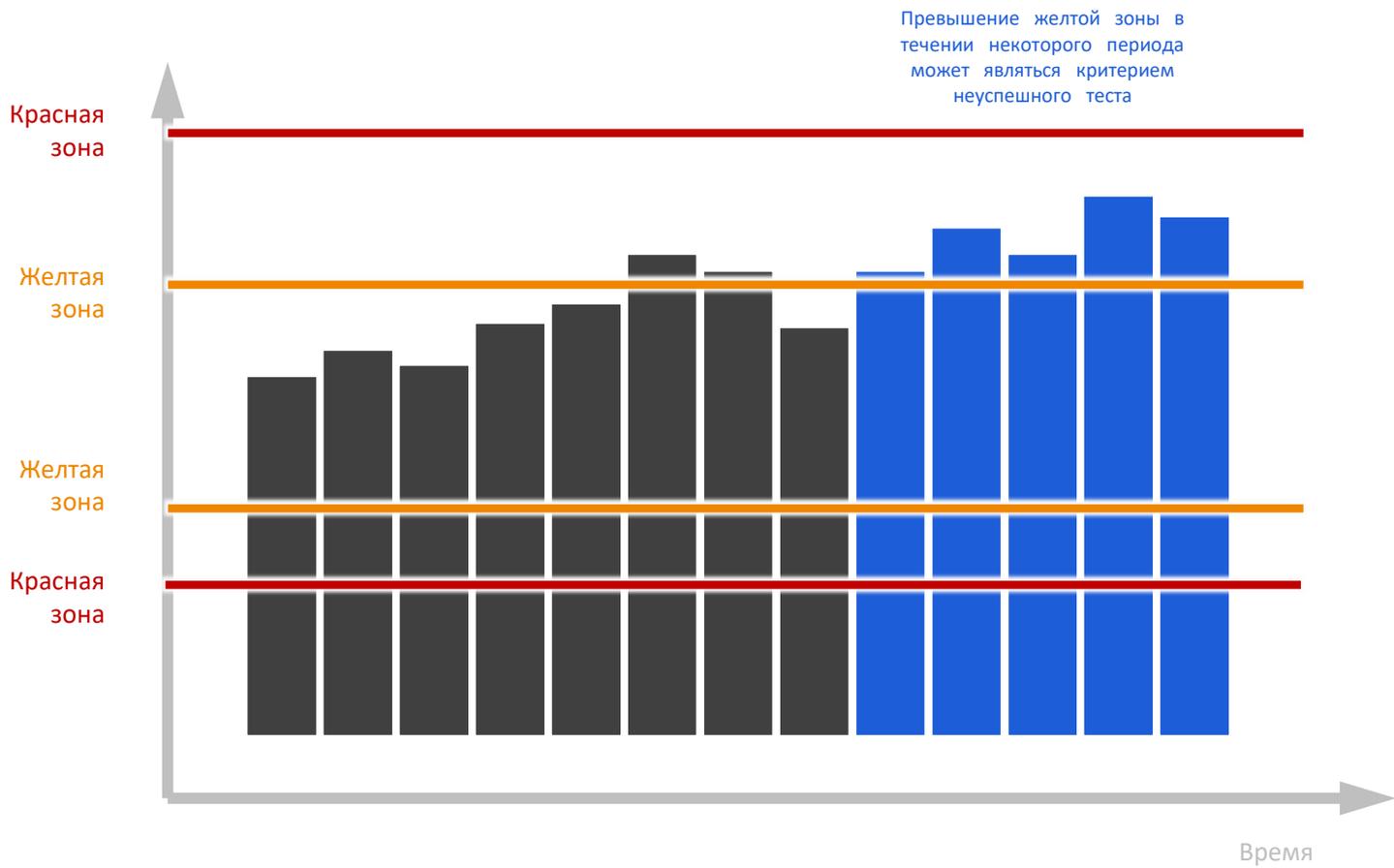
Светофор

проверок

- Требование не выполнено
- Пограничные значения
- Значения в пределах нормы
- Пограничные значения
- Требование не выполнено

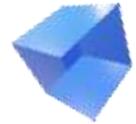
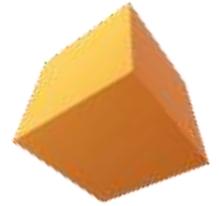
Суммирование желтых зон

1. Необязательность красных зон
2. Сложные критерии успешности
3. Итеративный подбор требований

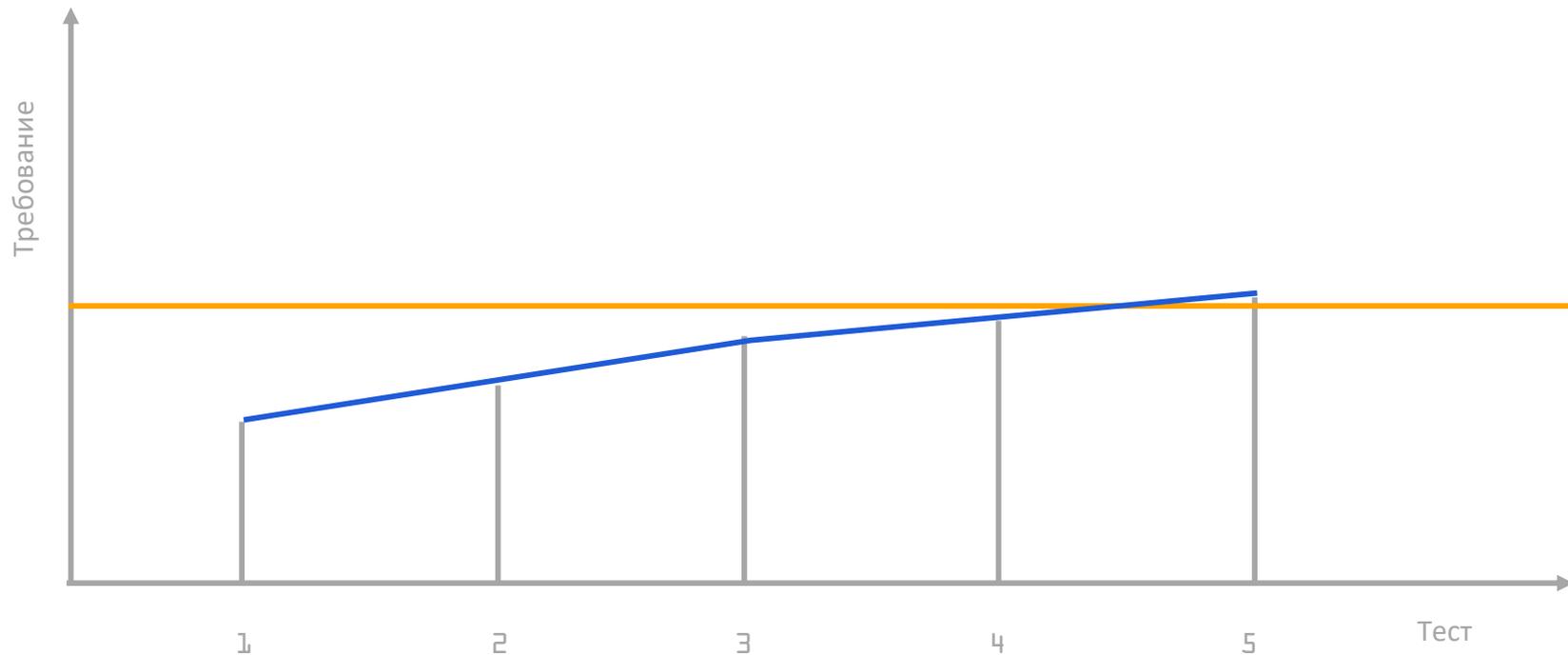


Абсолютные требования

1. Легко связываются с SLA/SLO
2. Просто в автоматизации
3. Просты в понимании



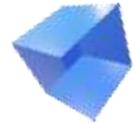
Проблема



Относительные требования

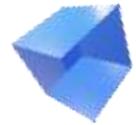
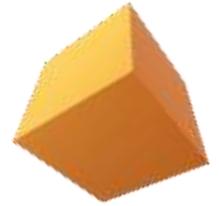


1. Защищают от пропуска трендов
2. Не заменяют абсолютные требования
3. Могут помочь при не фиксированной нагрузке



Оптимальные относительные требования

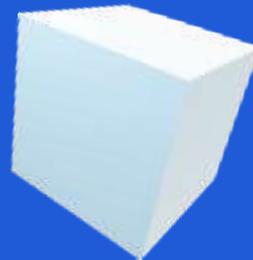
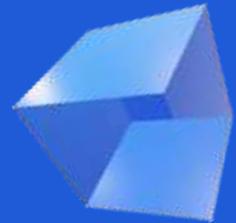
1. Время отклика
2. Сетевой трафик
3. Счетчик ретраев



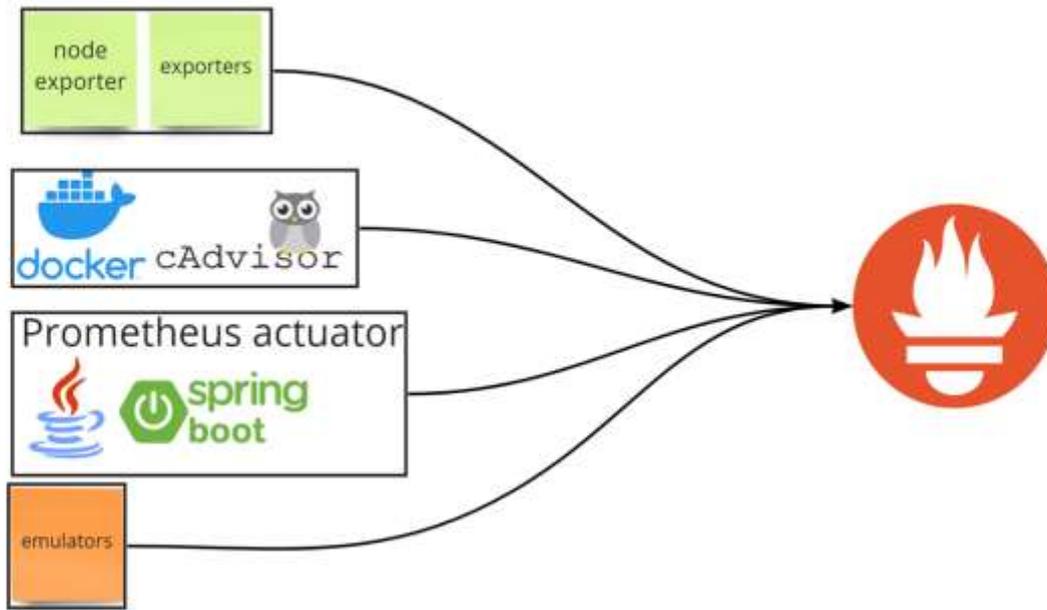
Повышает

однозначность

и доступность результатов



Данные для анализа



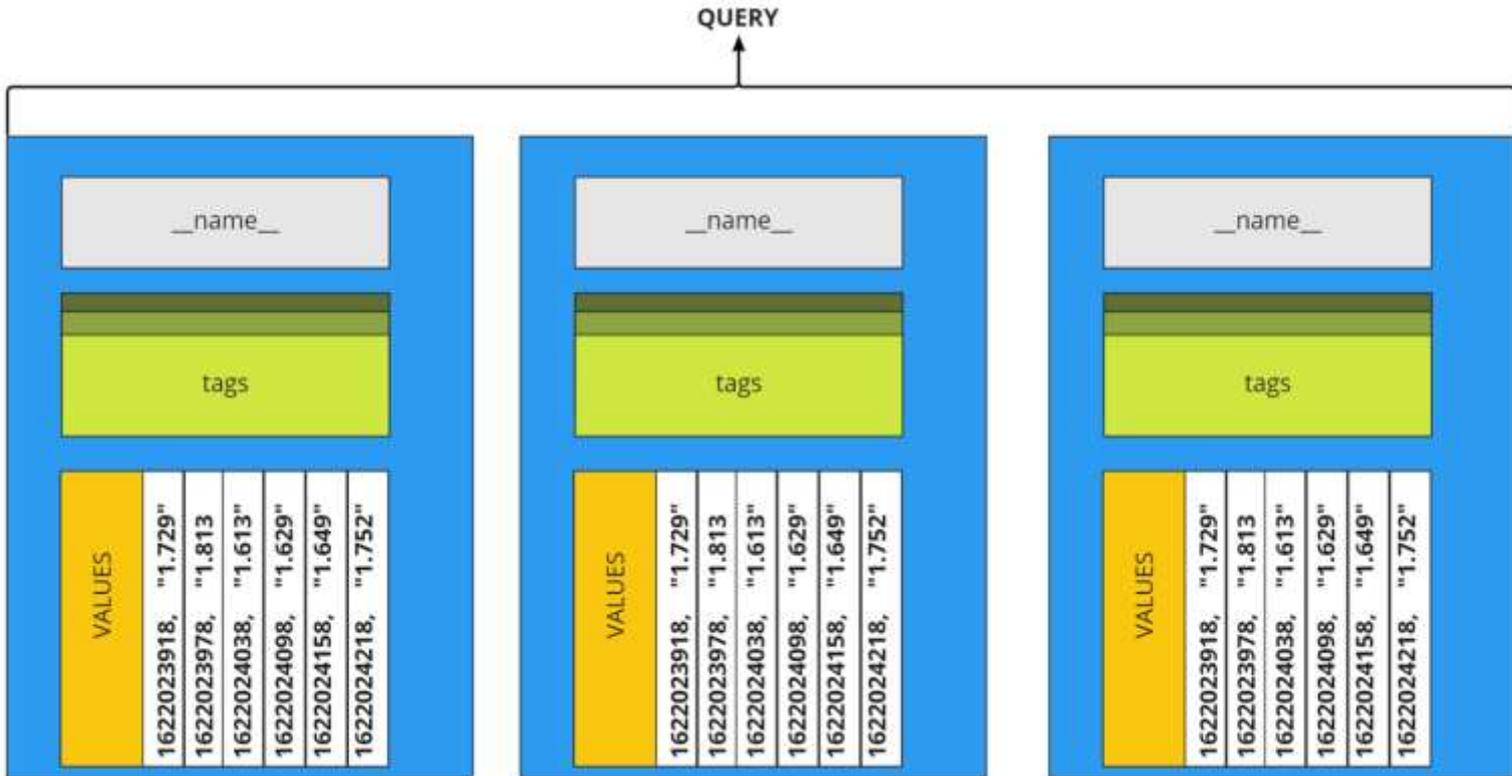
Http api prometheus

```
GET /api/v1/query_range
```

```
POST /api/v1/query_range
```

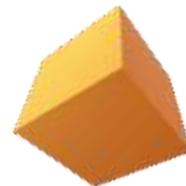


Структура



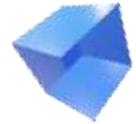
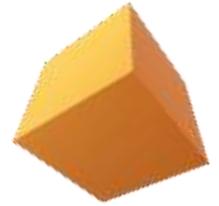
Шаблон запросов к prometheus

```
{
  "name": "success tps",
  "metricName": "sum(rate(bank_emulator_message_statuses_from_sbp_total{service=~\"(bank_emulator_perf-regress-2-env_dc2_1|bank_emulator_perf-regress-2-env_dc2_2)\",
msgStatusCode=\"100000\",msgType=\"C23\"}[1m]))",
  "attributesSource": "",
  "parametersCount": 0,
  "step": 60,
  "operations": [
    {
      "type": "AVG"
    },
    {
      "type": "QUANTILE",
      "describe": 95
    }
  ]
},
{
  "name": "error message statuses",
  "metricName": "avg by
(msgStatusCode,msgType)(rate(bank_emulator_message_statuses_from_sbp_total{service=~\"(bank_emulator_perf-regress-2-env_dc2_1|bank_emulator_perf-regress-2-env_dc2_2)\",
msgStatusCode!=\"100000\"}[1m]))\n",
  "attributesSource": "",
  "parametersCount": 0,
  "step": 60,
  "operations": [
    {
      "type": "AVG"
    },
    {
      "type": "QUANTILE",
      "describe": 95
    }
  ]
}
]
```



Локальный расчет статистик

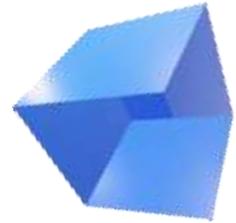
1. Учет пропущенных значений
2. Сохранение сырых данных
3. Подсчет нескольких статистик одновременно



Проблемы

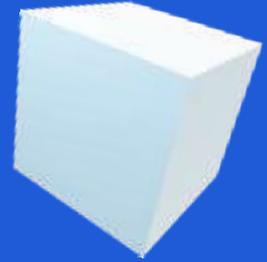
метрик

1. Много
2. Разрастание требований
3. Связывание метрик



Основные метрики

1. Связаны с требованиями
2. Коррелируют с поведением
3. Не зависимы



Во время теста **смотрим**
только основные метрики

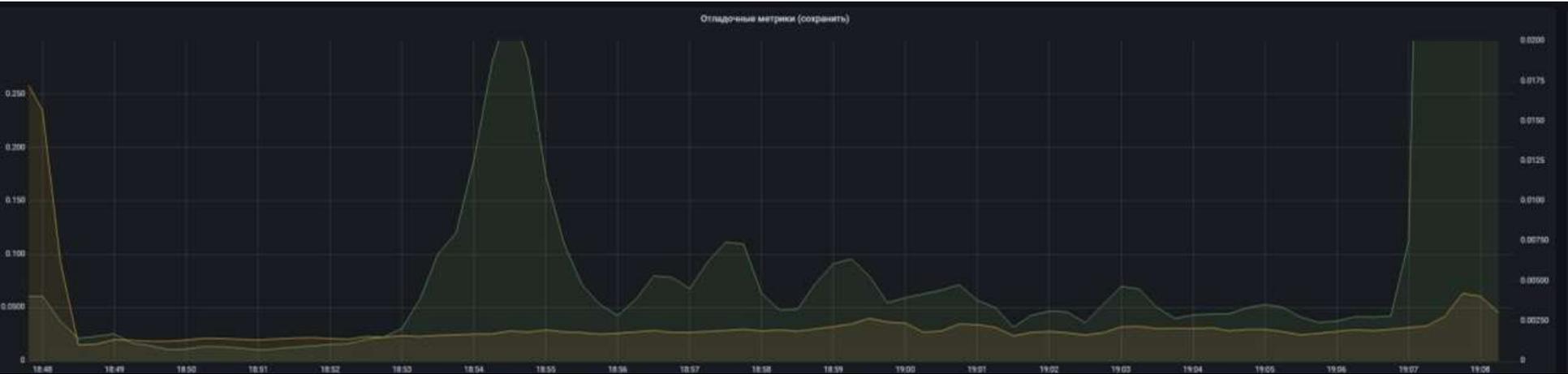


Хорошие примеры

1. Часто встречаются
2. Реактивно реагируют на проблему
3. Меняются быстрее других



Или вот так



Примеры ОСНОВНЫХ метрик

1. Уровень нагрузки
2. Суммарное количество ошибок
3. Время обработки верхнеуровневых транзакций
4. Хелфчек сервисов



Состояние

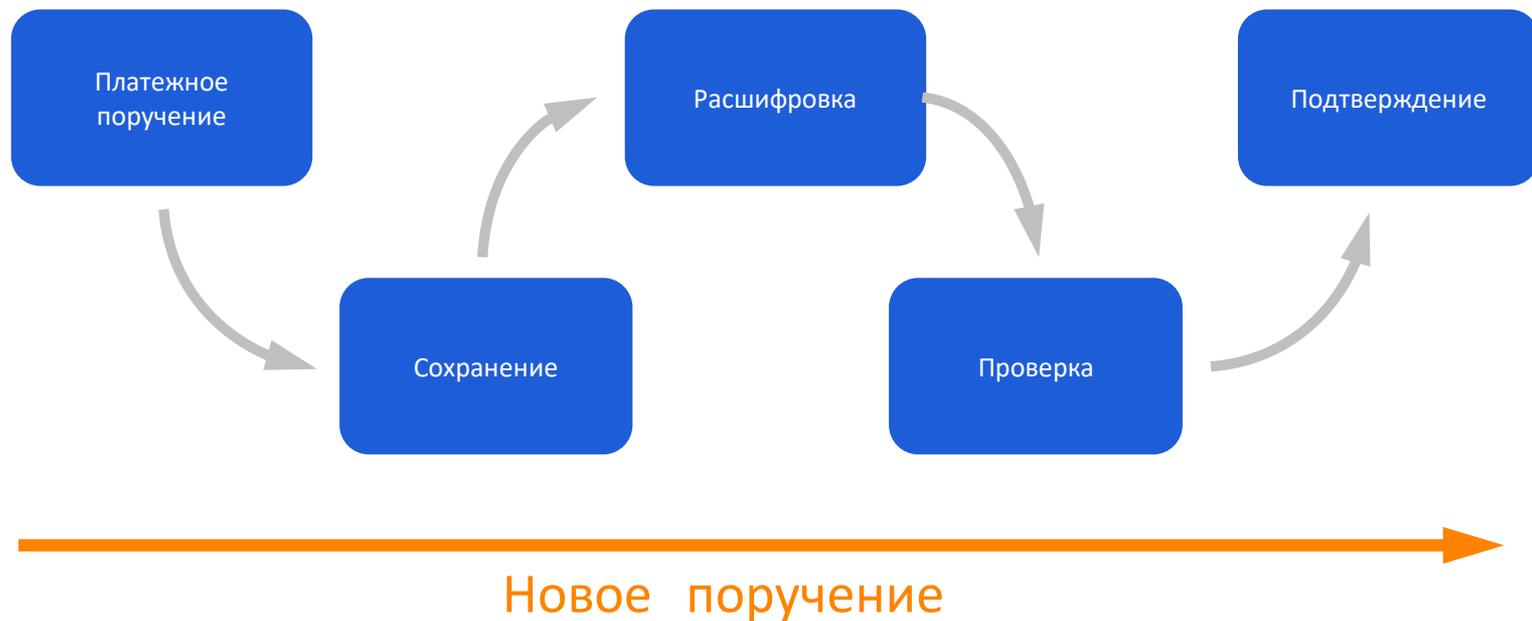
системы как метрика

1	1	1	1	1	1	1
1	1	1	1	1	1	1

Косвенные метрики

1. Расшифровывают основные метрики
2. Коррелируют с основными метриками
3. Локализуют дефекты
4. Иногда становятся основными

Косвенные метрики



Примеры

1. Рейт ошибок по кодам
2. Трафик по сервисам
3. ЦПУ по сервисам и по серверам
4. Времена отклика на отдельных элементах
5. Уровень нагрузки на отдельных элементах

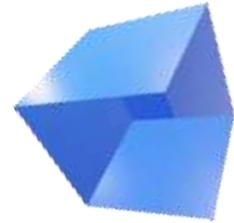
Сколько каких метрик нужно

1. 10 - 15

ОСНОВНЫХ

2. По 2 - 10

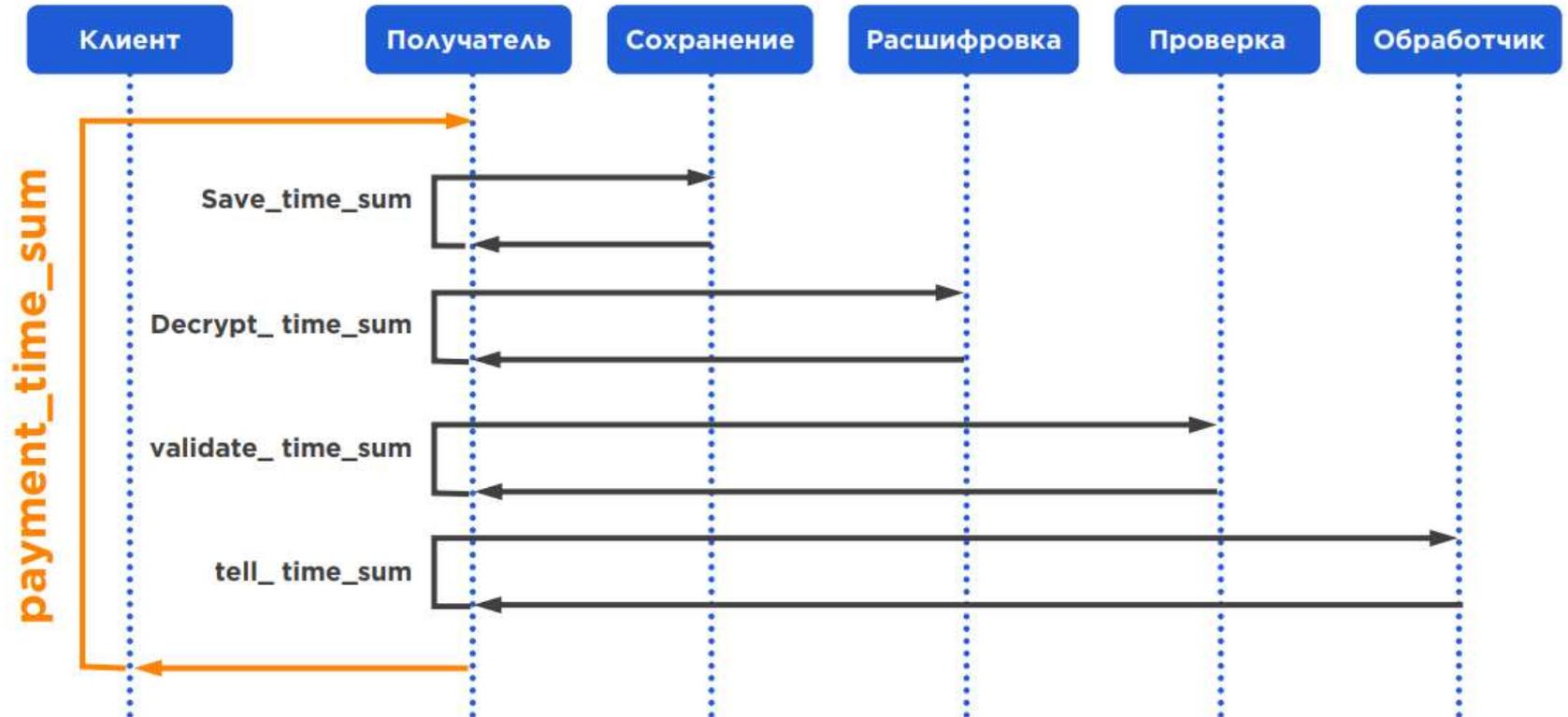
КОСВЕННЫХ на одну основную



Собственные метрики

1. Точки распределения запросов
2. Инъекции в коде
3. Эмуляторы

Micrometer



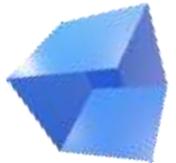
Пример

```
Timer decryptTime = registry.timer("decrypt_timer", Tags.of("service", "receiver",  
"message_type", "x1", "operation", "decrypt"));
```

```
Instant startHandle = Instant.now();
```

```
val decryptedX1 = cryptoService.decrypt(x1);
```

```
decryptTime.record(Duration.between(startHandle, Instant.now()));
```



Асинхронный пример

```
Timer decryptTime = registry.timer("decrypt_timer", Tags.of("service", "receiver",  
"message_type", "x1", "operation", "decrypt"));
```

```
Instant startHandle = Instant.now();
```

```
CompletableFuture future = cryptoService.decrypt(x1);
```

```
future.thenAccept(() -> {
```

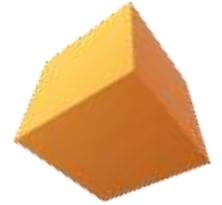
```
...
```

```
decryptTime.record(Duration.between(startHandle, Instant.now()));
```

```
})
```

Интеграция со спрингом

1. Стартер
2. JVM включен
3. Интегрирован с другими модулями спринг



В актуаторе

decrypt_timer_seconds_count{operation="decrypt",message_type="x1",service="receiver"} 143669.0

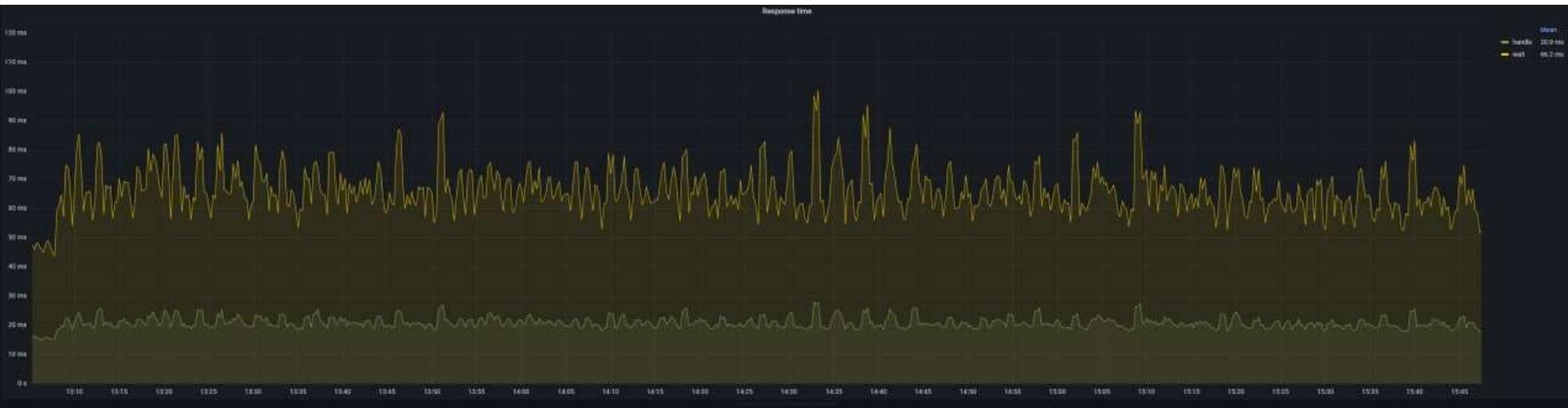
decrypt_timer_seconds_sum{operation="decrypt",message_type="x1",service="receiver"} 56043.1037186



Скорость принятия решений и однозначность результатов



Обработка регулярной высокой дисперсии метрик

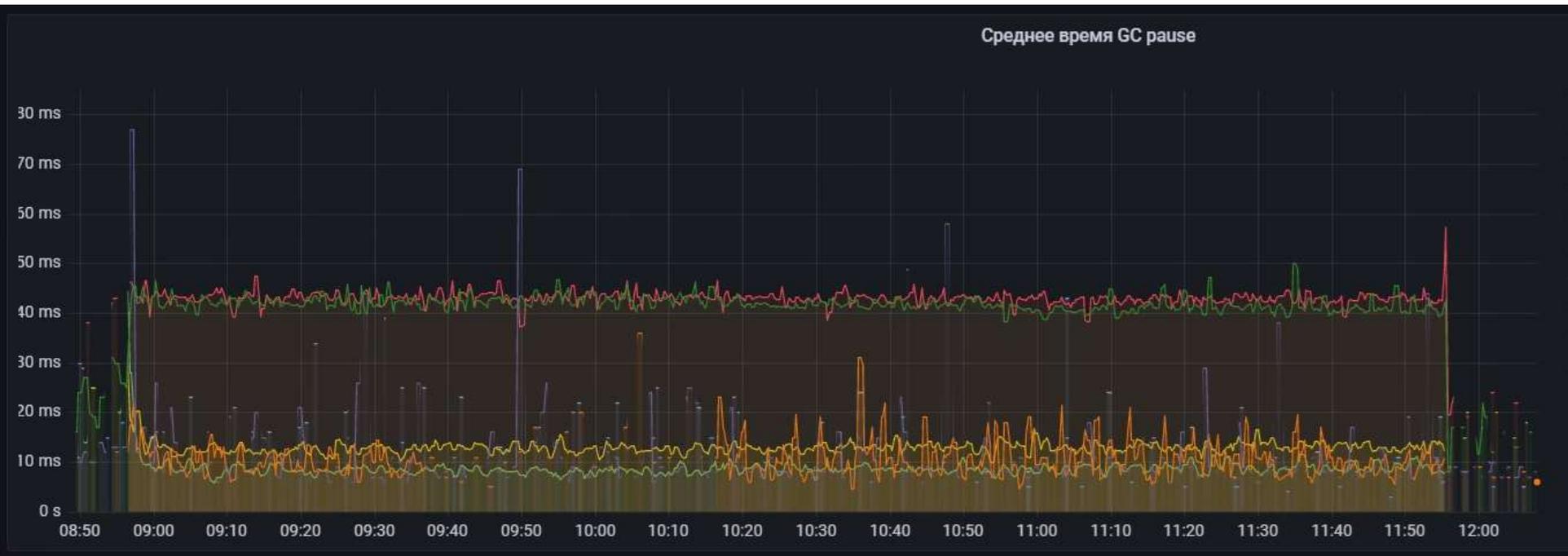


Лучше пройти мимо

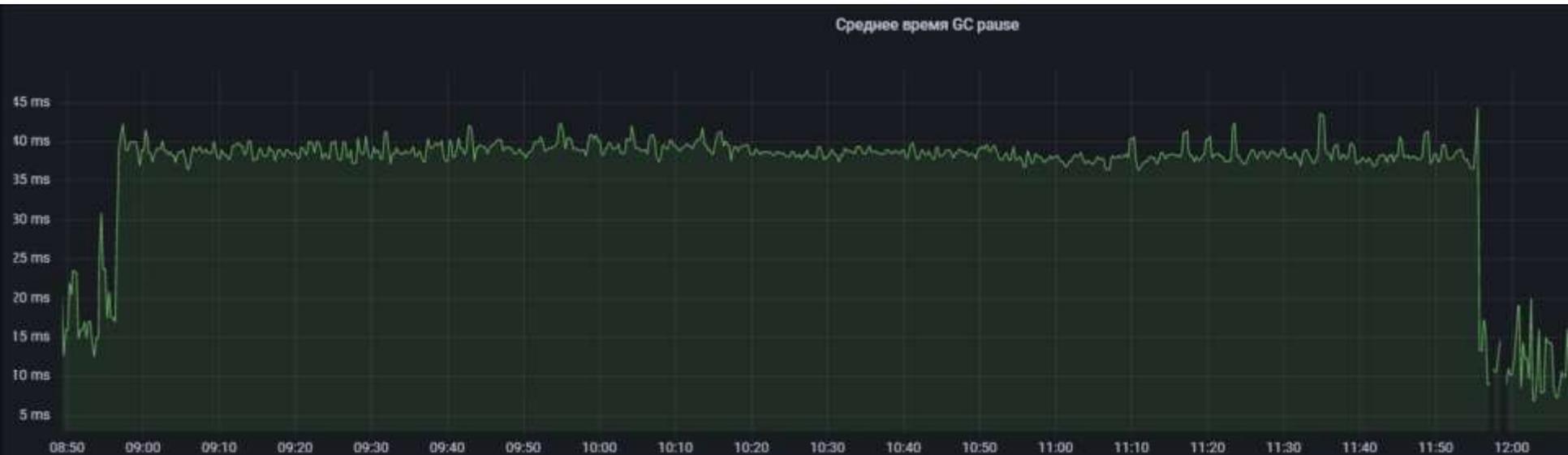
1. Сложная модель поведения
2. Не достаточная реактивность
3. Слишком высокая реактивность



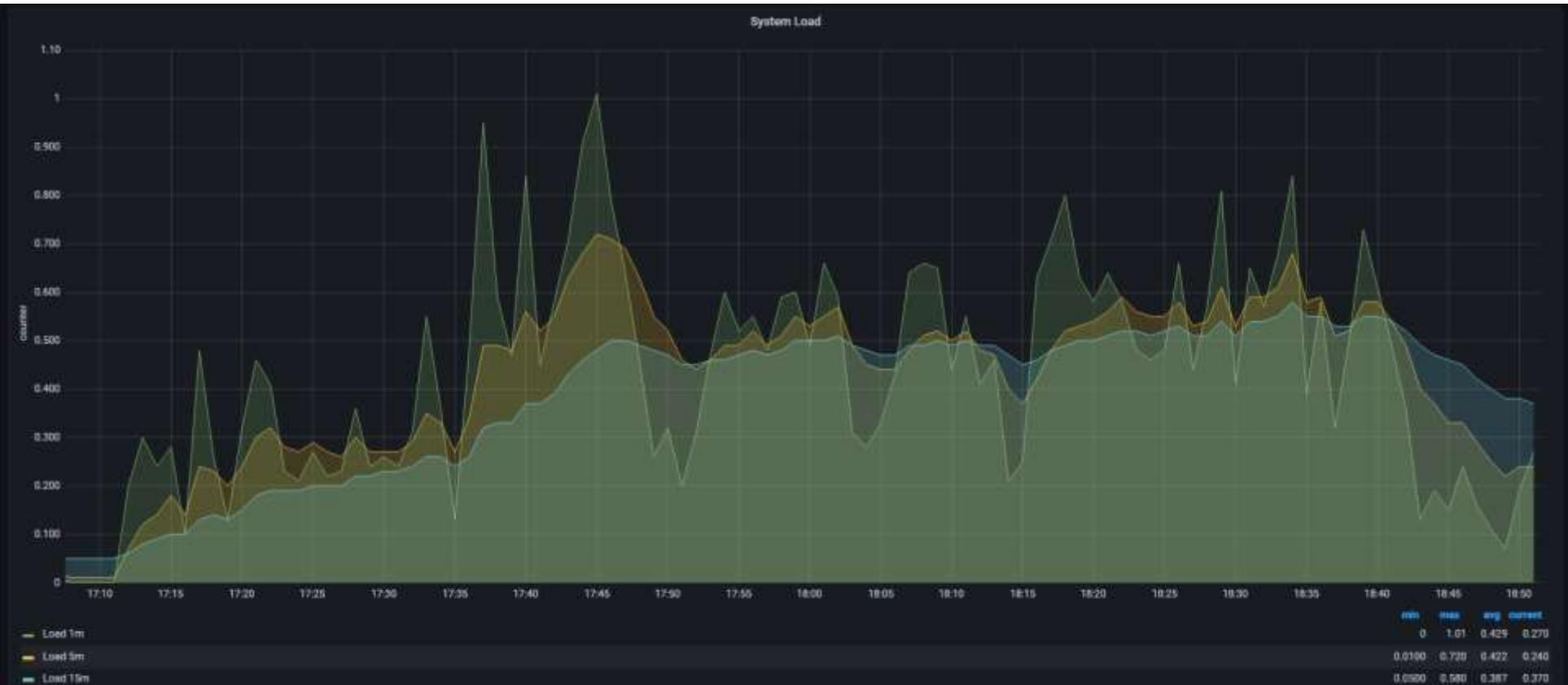
Метрики по сервисам



Метрика суммы



Сглаживание



Счетчик условий

1. Удобно когда большой разброс
2. Накапливается со временем

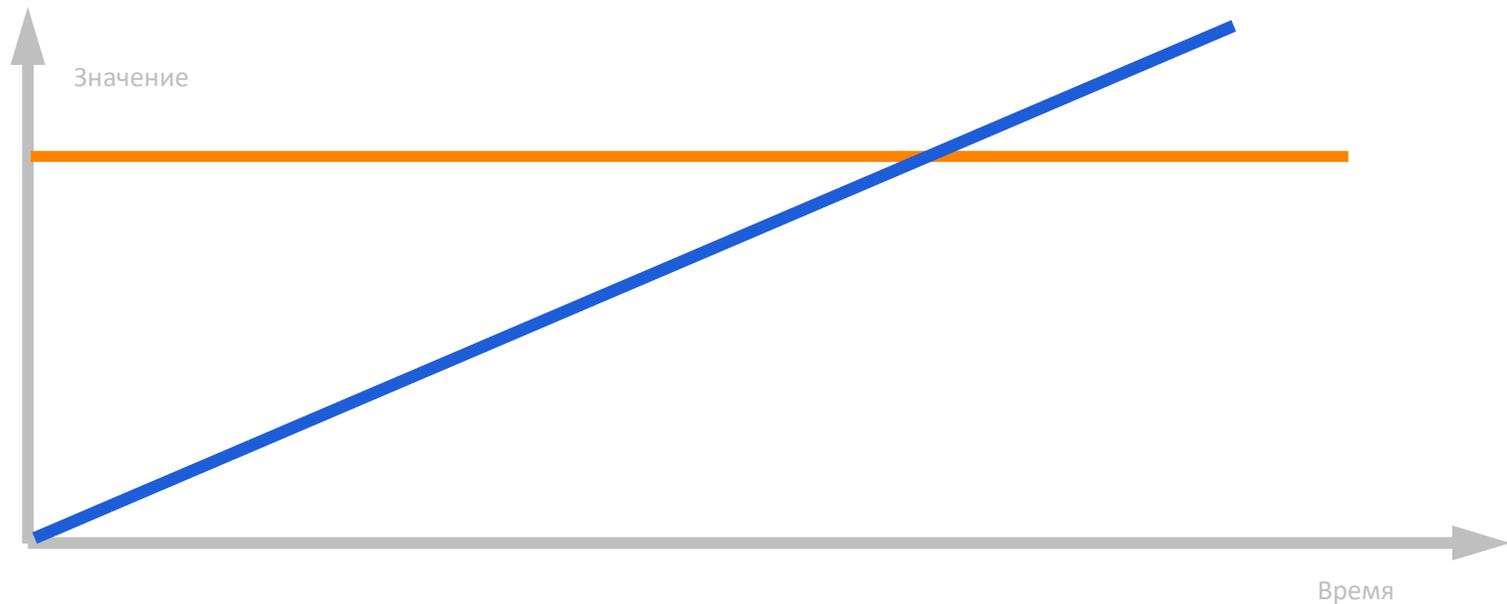


Сильносвязанные метрики

1. Работа GC, финализаторы и таймауты в системе
2. Нарастание очереди и снижение уровня нагрузки
3. Таймауты и ошибки
4. Синхронные запросы и нарастание оперативной памяти

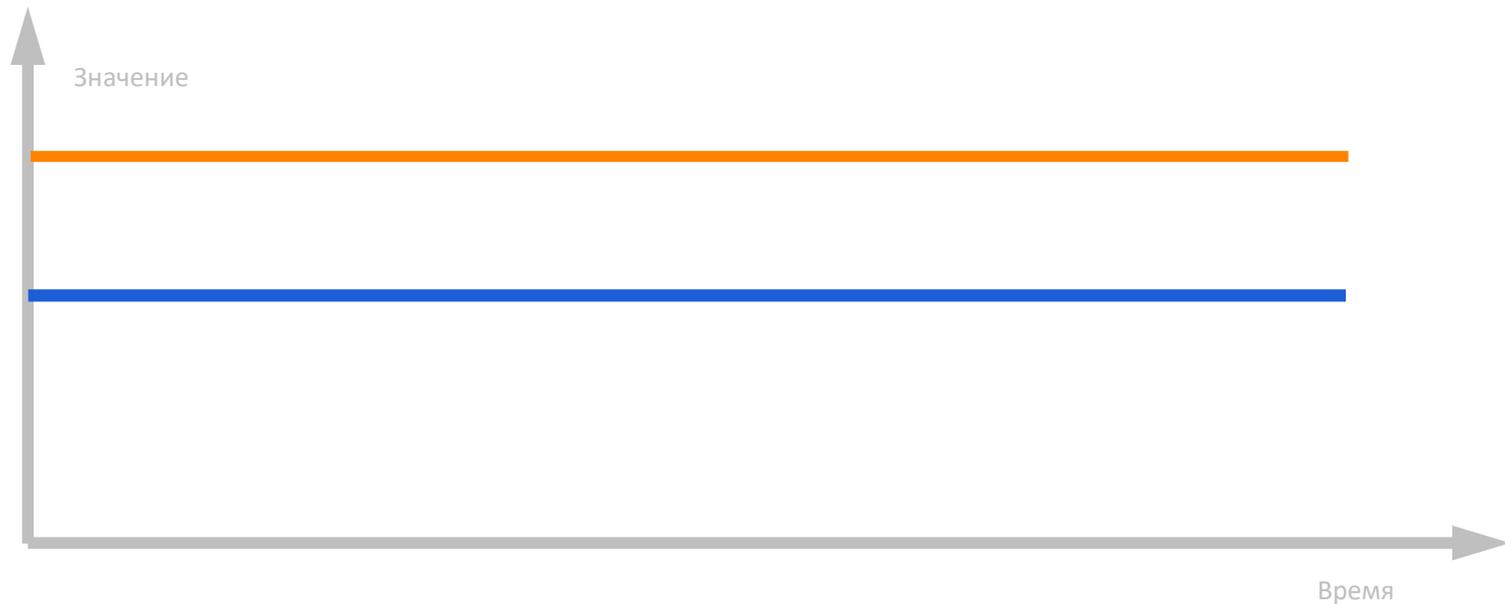
Сырая метрика

Фиксация уровня метрик

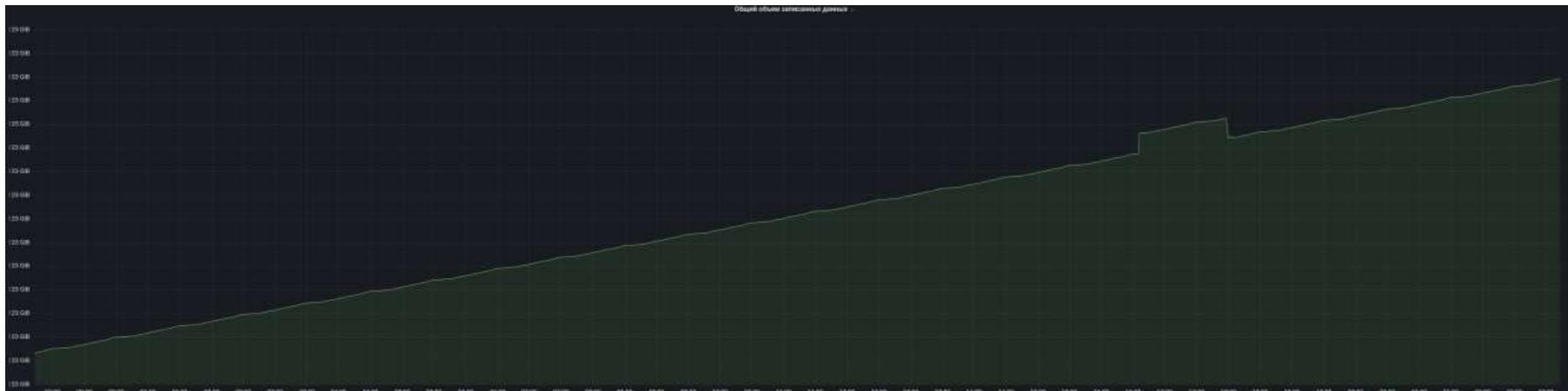


Производная

Фиксация уровня метрик



Фиксация уровня метрик • Сырая метрика



Фиксация уровня метрик.

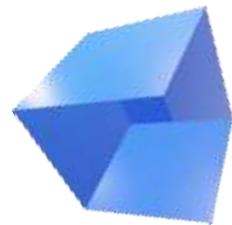
Производная метрика





$$\frac{\sin x}{n} = ?$$

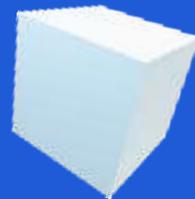
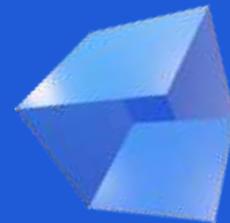
$$\frac{\sin x}{n} = ?$$



$$six = 6$$



Однозначность и доступность



Определение связности метрик

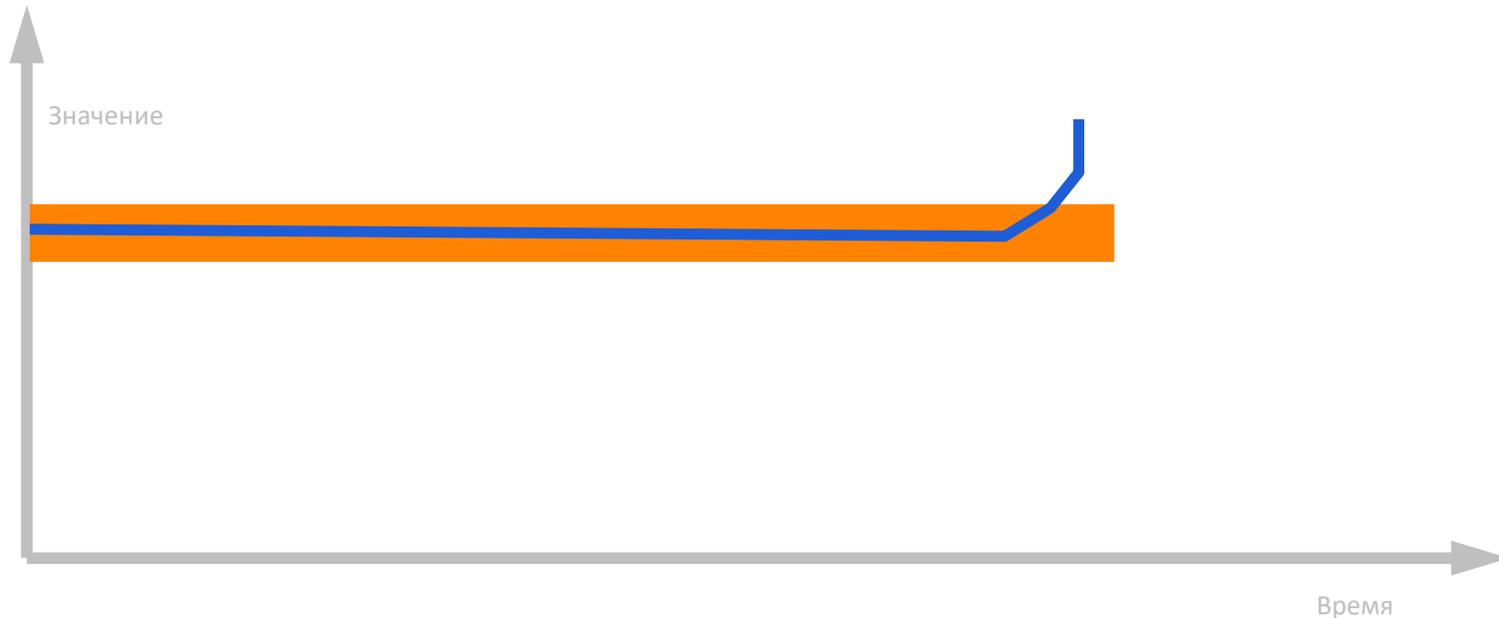


Варианты построения цепочек

1. Статистические
2. Аналитические



Продвижение по цепочкам



Уровень нагрузки бизнес-процесса

1. Время обработки бизнес-процесса
2. Ошибки по бизнес-процессу

Время обработки транзакции

1. Времена отклика элементов транзакции
2. Ресурсы связанной системы



Утилизация CPU

1. Пропускная способность операций
2. Ресурсы сервера



Скорость и доступность

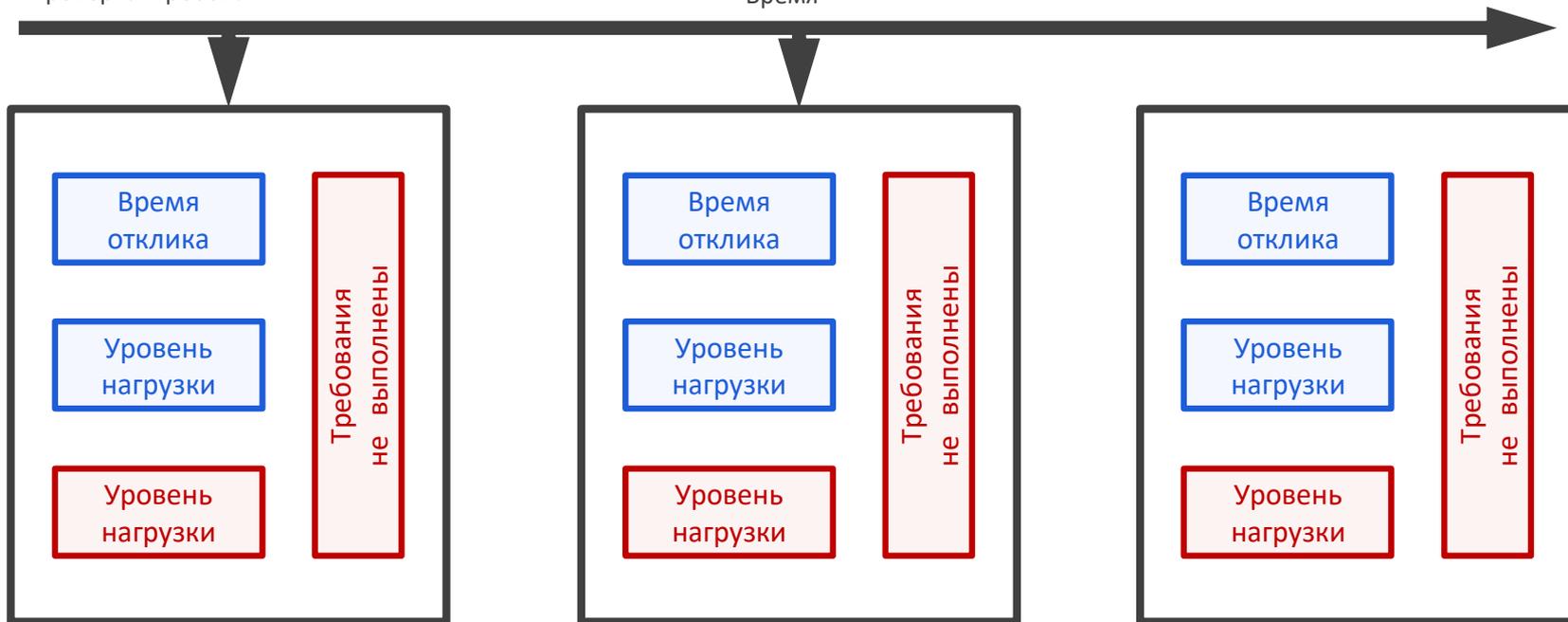


Обработка ЛОЖНЫХ срабатываний



Проверка требований

Время



Останавливаем тест

После остановки

1. Сбор метрик
2. Поиск корреляций
3. Отчет

Тест остановлен

Релиз: 1.66.3

Стенд: perf-1-eln

Parent-uid: e30f01f8-73c8-44f8-974b-56be2d06a238

UID этапа падения: 54aac763-5903-40ae-afde-e4f5a8f1f751

Наименование этапа: —

Время начала теста: 07.03.2021 16:35:47

Длительность теста: 3922 seconds

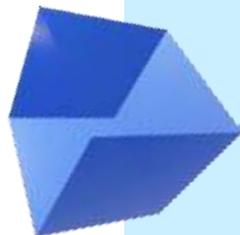
Время от начала этапа: 1522 seconds

Ссылка на [графану](#):

Наименование метрики	Значение	Требование	Зона
Errors	47	4	Red

Связанные метрики

Наименование метрики	Коррелирующая метрика
Errors	—



Отчет

о неуспешном запуске



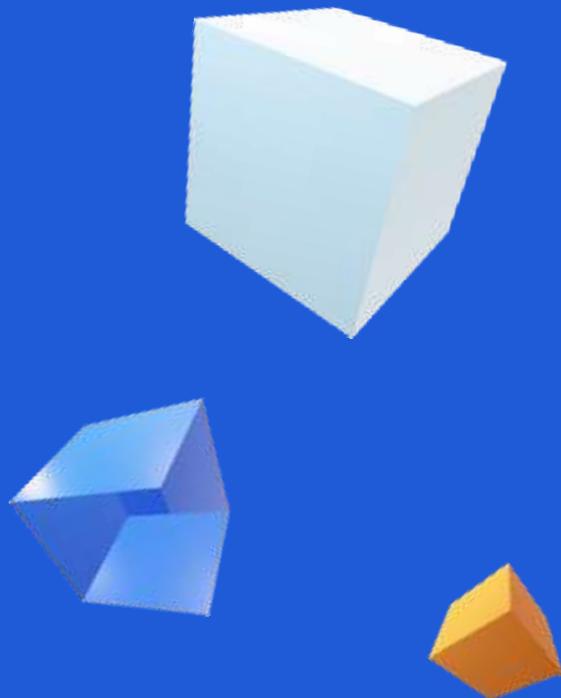
1. Нарушенные требования
2. Этап тестирования
3. Связанные косвенные метрики

Тегирование результатов

1. Возможность фильтра
2. Подбор эталона
3. Группировка по тэгам

Подведем итоги

1. Начинаем с регресса
2. Простые сценарии
3. Настройка требований
4. Подбор метрик
5. Обработка метрик
6. Собственные метрики
7. Деревья связности метрик
8. Кэш с текущими метрикам

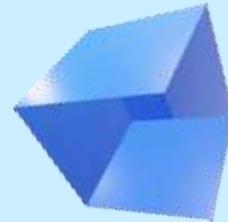
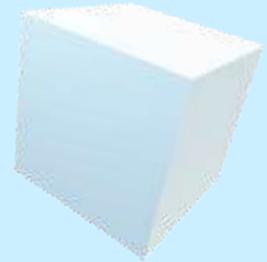


Другие виды тестов

1. Сложнее вывод
2. Меньше накопленных данных
3. Сложнее модель поведения

Подведем итоги

1. У проверки качества есть свои метрика качества
2. Для формирования базиса нужен "*perfect vision*"
3. Улучшение качества дает преимущества
4. Автоматизация остановки тестов требовательно к качеству процесса

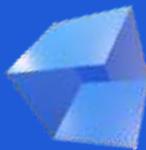


Спасибо за внимание!



Дмитрий
Цитман

Инженер нагрузочного
тестирования Системы Быстрых
Платежей -
Команда Мир Plat·Form



HEISENBUG